To be published in
IEEE Trans. on Nucl. Sci.

Invited paper to be presented
at Nuclear Science Symposium,
San Francisco, Nov. 14-16, 1973.

BNL 18360

Conf- 731112-- 35

FUNCTIONAL DISTRIBUTION - AN ARCHITECTURE FOR MULTI-USER COMPUTER NETWORKS IN INSTRUMENTATION*

D. G. Dimmler
Brookhaven National Laboratory
Upton, New York 11973

November 1973

*This work was performed under the auspices of the U. S. Atomic Energy Commission.

BNL 18360

FUNCTIONAL DISTRIBUTION - AN ARCHITECTURE FOR MULTI-USER COMPUTER NETWORKS IN INSTRUMENTATION*

D. G. Dimmler

Brookhaven National Laboratory
Upton, New York 11973

ABSTRACT

A new approach is introduced for the design of Multi-User Real-Time Computer Networks. The approach leads to a system architecture called "Functional Distribution". Definitions of the concept and key issues of a practical implementation are discussed using as an example a Multi-User Experiment Control System. The discussion also includes the architecture of that hardware which has been traditionally referred to as the "interface between an experiment and a computer".

A. Background

In the late 50's, it first became feasible to use real-time computer installations in instrumentation and laboratory automation. The key question to be answered by the system designer at that time was:

> How to use these new real-time computers to enhance the performance of experimental or laboratory setups?

In the Nuclear Instrumentation field, the problems attacked were mainly associated with multiparameter, multichannel analysis.[1,2]

Developments in the computer industry rapidly modified the question to:

> What is the most appropriate computer architecture for a given laboratory environment?

Pursuance of this question brought a confusing variety of approaches towards the late 60's[3,4,**] in the area of multi-user systems.*** Several attempts at classification of the approaches have been made, however.[5]

Dedicated architectures had the reputation that they could be used independently of other people's plans, experiments and laboratories. The tradeoffs were in the economy of high-performance peripherals which were too expensive to connect to dedicated architectures and in higher level languages which were geared towards more general configurations. With the intent to overcome the above limitations, support grew for general-purpose multi-user centralized architectures where all resources were pooled at one structure with the idea of gaining economy and performance by sharing with some tradeoffs in independence flexibility.

Disappointments in economy, flexibility, independence and sometimes performance of many general purpose centralized architectures soon shifted the major interest to networks where the system designer was faced with still another question:

> How can the tasks requested from the system be partitioned onto the various components of the network?

Partitioning, a familiar task for computer hardware designers, hardware interface designers and computer software designers, now entered system design. Partitioning gained further significance with the continued introduction of more and more complex functions by LSI circuit manufacturers. The LSI functions introduce partitions which have the tendency to make both the familiar hardware partitions and the independent software partitions uneconomical. Some combinations of the above and other parameters**** may account for the confusing presence of network architectures.

I was recently asked to propose a new architecture for the replacement of a one-decade-old dedicated multiple-experiment system[6,*****] with a more modern facility.****** Besides performance and other improvements, the following general features are expected from the new facility:

**References from Nuclear Instrumentation are given here. There is ample evidence that the situation has been similar in the general field of instrumentation and automation.

***Multi-user system is here defined as a system which operates several experiments or laboratory setups concurrently.

****Political boundaries, organizational boundaries, funding regulations, etc., have, of course, a substantial influence here. These parameters, although implicitly considered, are not within the scope of this presentation.

*****The present facility to be replaced is called: Multiple Spectrometer Control System (MSCS).

******The new facility is called: Reactor Experiment Control Facility (RECF).

*This work was supported by the U. S. Atomic Energy Commission.

129

(a) A single high-level application language which affords maximum convenience to the user should exist in order not to load down the experimenter with complex computer terminology.

(b) Protection against accidental interaction between experiments should be increased to a degree that experiment A can be in a testout and modification phase, whereas experiment B is in an unattended routine run phase without unscheduled interactions.

(c) The facility should be incrementally expandable with prevailing state-of-the-art hardware and software. This expansion is to apply to new experiments with presently unknown specifications as well as to existing experiments with changing specifications.

(d) The previous point implies that parts of the facility may become obsolete in the future. The facility as a whole, however, should not have a scheduled obsolescence.

In searching for material I came across a rhetorical question mentioned in a summary of a panel discussion in 1969 on "Future Trends in Nuclear Instrumentation".[7] The question was: "Whatever happened to the (promised) flexibility (of on-line computers)?" The answer was: "It simply did not happen".

Instead of finding an answer in studies of newer systems, I was prompted to other questions: "Why are system architectures becoming obsolete faster than the hardware and software components they incorporate and are made of?" "Is there something wrong with the traditional approaches to system design?"

I would like to shed some light on these questions in the presentation. In the following, an approach to system design is introduced which has led to a system architecture called Functional Distribution. Some key issues of the approach and its implementation are discussed. Although the discussion of the approach is based on the example of a system used in the control of multiple experiments, it is believed that the approach as well as the results have general applicability to real-time system design.

B. Analysis of the Approach

The approach is based on two conclusions:

(a) Experimenters as well as experiments relate to an experiment control system* on the functional level. Typical requests issued by an experimenter are: prompt experiment parameters, start experiment, move motor from position A to position B, stop experiment. If the experiment is involved in the analysis of data, then a typical command to the system is: move data from File A to File B.

(b) One of the major mistakes of traditional approaches has been to partition, too early, a given functionally defined task to be performed into a hardware task and an independent software task. Quite often this partition has been a priori assumed.

Figure 1 shows this partition into hardware and software tasks using as an example the function "BINARY-TO-BCD-CONVERSION" which might be necessary for an expansion of an existing system. This elementary Functional Subsystem is embedded somewhere within a Functional Architecture and is partitioned into two implementation levels. These two levels are the following:

(a) A set of Software Subsystems consisting of programs, including instructions and algorithms. These programs are embedded in Software Structures.

(b) A set of Hardware Subsystems, defined in more detail as Resources,[10] are embedded into Hardware Structures.

The example shows the dilemma of the a priori partitioning. LSI technology provides, as of recently, another implementation partition which is: to use an LSI circuit[11] and to forget about the software structures as well as a fair amount of the hardware structure and embedded subsystems. This, however, can only be done if a functional architecture is recognizable. This example shows, in addition, that complex hardware structures and associated software structures become technologically obsolete because a device having a cost of a few dollars appears and seems to survive on the market.

*A system is defined here as follows: A system has its own environment and is, in fact, a subsystem of some broader system.[8,9] A system may be broken down into an embedded set of more elementary subsystems. An Experiment Control System is more specifically defined here as a real-time system used in the control of, data acquisition from, and monitoring of experiments.

In order to avoid the above mentioned serious drawbacks, the following strategy in the design of a new system seems to be appropriate:

(a) study the system to be designed on a functional level;

(b) break down the system into a set of hierarchical, nested functional subsystems which are interconnected via a Functional Architecture;

(c) then, at some point which looks reasonable, partition the functional subsystems into independent hardware and software subsystems.

Iterations within the strategy are, of course, necessary because it is necessary (a) to use as many existing implementation structures as are economically and otherwise feasible and (b) to find architectures which hopefully fit into future trends of partitioning in the computer and electronics industry.

Before I proceed, the definition of a node should be introduced.

A node is a functional subsystem* which has, on the implementation level, an embedded hardware structure** and, if applicable, an embedded software structure. The node boundary is defined in functional terms, called a protocol.*** Within this definition, a typical general purpose Centralized Resource Sharing system would constitute one node; a set of dedicated systems would constitute a set of nodes which have no architectural interconnection and a network consists of a set of nodes which are incorporated within an architecture of the system. The node, of course, may include an internal architecture. In Fig. 1, the two functional subsystems FS1 and FS2 are members of one node.

C. System Overview

Experiment Control Systems are organized along Data Flows and Control Flows**** as shown in Fig. 2. Three environments can be recognized:

*A functional subsystem may consist of a set of more elementary functional subsystems which are embedded.

**It follows that a node includes at least two hardware resources: a processor and a memory resource.

***The protocol may, in turn, include an embedded hardware structure and software structure.

****This is a noteable difference to some other systems. For instance, traditional time sharing systems are procedure oriented.

(a) the Experiment Environment, which is self-explanatory;

(b) the Input/Output Environment, where all input and output media, such as cards, magnetic tapes and/or connection to a larger computer center, are managed;

(c) the Control and Monitoring Environment, which includes the control devices, usually teletypewriters and function keyboards as well as monitoring display devices.

A Data Flow exists between the Experiment and Input/Output Environments. The Main Control Flow is directed from the Control Environment to selected points within the Data Flow. Monitoring information flows in the opposite direction of the Main Control Flow and originates from arbitrary points within the Data Flow.

D. Global Subsystems

As shown in Fig. 3, section 1, an Experiment Control System may be broken down into global functional subsystems which, in turn, may be further broken down into more elementary embedded functional subsystems.

The Experiment Service Subsystem includes hardware and software for experiment-related functions such as motor control, control of data transfer from and to the experiment, local processors within the data flow, etc.

The Application Subsystem most closely resembles the traditional general purpose computer with a general purpose computer processor and memory on which application programs***** are operating. In the case of experiment control, this program resembles the main control, the measuring sequence and associated analysis. This subsystem is not discussed further here.

The File Subsystem includes the hardware and software for the management of on-line files on file storage devices.

The I/O Subsystem includes the hardware and software for the peripheral devices and/or for transmission to a larger computer center. Data Acquisition Systems and Monitoring Systems are, as seen in Fig. 3, simpler subsets of the Control System.

E. Properties of System Design Parameters

Known implementation properties of some system design parameters are shown in Exhibit 4. Elaborations on the parameters are given in Appendix A. It is the purpose of the exhibit to assist in assigning functional

*****Application programs, as opposed to system programs and library programs, are generally written in a high-level language.

subsystems to nodes so that the various subsystems can be economically implemented with the least complexity with consideration given to the possibility of using existing designs and predicting future trends.

Computer installations always represent a major monetary and organizational commitment.* Therefore it has been decided to group the parameters according to their primary impact on cost.** The exhibit shows that three cost areas must be considered:

Investment cost

Operating cost

Depreciation cost, where only the influence of technological obsolescence is of interest here.

Key considerations in the assignments are:

(a) Reasonable tradeoffs among investment cost, operating cost and depreciation cost of a node (not the entire system) are attempted.

(b) Isolation of malfunctions[10] (isolation against unintentional interference) between experiments and between functional subsystems should be as good as is presently economically feasible.

(c) It is the intent to find protocol definitions which have a substantially lower rate of technological obsolescence than the functional subsystems themselves. This way, functional subsystems can be replaced selectively and updated without rendering the system obsolete.

(d) An important consideration is the present feasibility of partitioning because software offered by most manufacturers in the computer field is based on centralized processor and primary memory sharing principles. It is believed that economic considerations will prompt computer manufacturers to make software generally available which is based on distributed processor principles.

*Organizational commitments are often expressed in terms of cost.

**Secondary and iterative impacts are assumed to be minor and are not considered here.

F. A Practical System

Assignments

A study of the parameter properties leads to the subsequently described assignments.

The Input/Output Subsystem and File Subsystem for all experiments are incorporated into one node and are shared as shown below. The reasons are the following:

(a) Economic reasons and, in the case of the File Subsystem, logistic reasons furnish an incentive for sharing.

(b) The functional boundaries are fairly solid*** due to the relatively low sophistication necessary for experiment control systems.

(c) The mean time between intentional changes is long. Thus the node has a chance to be left alone for long periods of time.

(d) The technical obsolescence rate of the subsystems depends on input/output devices and file devices. These resources become obsolete at a low rate. Thus there is a good chance that the node has a long, technologically useful life expectancy.

(e) The node is generally applicable in fields unrelated to real-time applications. Thus, the probability of future gain in economy is enhanced.

(f) The required response times are long compared to the speed of the central processor and the primary memory. Thus, efficient scheduling of these resources for the purpose of sharing is feasible.[12]

The Application Subsystem as well as part of the Experiment Service Subsystem of each experiment will be incorporated into nodes which are private to the experiment. The reasons are as follows:

(a) Sharing of processors and primary memory becomes inefficient because of the short response time usually required.[12]

***The most solid boundary in the computer field seems to be the functional boundary between the I/O subsystem and the I/O environment. Card formats, magnetic tape formats and protocol change very little. The artificial differences which can be often found resemble more a degree of confusion rather than a difference in need.

(b) The decreasing relative cost of processors and primary memory makes sharing for economic reasons unattractive. There are no other reasons for sharing.

(c) The independence between experiments which is gained by private nodes has beneficial effects in the areas of: (a) protection against accidental interactions, and (b) future incremental expandability with prevailing state-of-the-art technology.

G. Global Architecture of Network

Figure 5 shows the global architecture of the network being developed. A Private Control Node, which includes the appropriate Application Subsystem and part of the Experiment Service Subsystem, is assigned to each individual experiment. The Main Control devices, usually terminals of some sort, are connected to the appropriate node. The Private Control Nodes interconnect to Service Nodes in both directions within a hierarchical architecture. The <u>Main Control Flow</u> of the system originates at the Main Control Devices and is directed <u>from</u> the Private Control Nodes <u>to</u> the service nodes. Service Nodes shared by several Control Nodes are located in higher levels of the hierarchy. In Fig. 5 one Central Shared Service Node is shown. Service Nodes which are accessed exclusively by one Private Control Node are located on lower levels of the hierarchy and are referred to as Private Service Nodes.* Command and data transfers within the system operate on a request/response basis. Command transfers in the direction of the Main Control Flow, which is shown in Fig. 5, are considered to be <u>Service Requests</u>; command transfers in the opposite direction are considered to be <u>Service Responses</u> to previously given requests.

H. Central Shared Service Node

We presently plan to combine three elementary Input/Output Subsystems and one File Subsystem into one <u>Central Shared Service Node</u> as shown in Fig. 6.

The <u>Reserved Device Subsystem</u> includes the devices and appropriate management for datasets** which are explicitly reserved and released by Control Node programs. These include the familiar unit-record devices such as magnetic tape units, card reader, and paper tape reader/punch.

*On the hardware implementation level, each Control Node contains, for logistic reasons, a general purpose computer processor. The Shared Service Node contains, for practical reasons, a general purpose computer processor. The Private Service Nodes often do not contain a general purpose computer processor.

**The definition of a dataset here depends on the target subsystem. It is roughly equivalent to a device in the Reserved Device Subsystem and to a file in the File Subsystem.

The <u>Spooled Output Subsystem</u> includes the output devices and management for the devices which are operated in a spooled fashion. The Control Node program submits output data to the Subsystem which holds the data temporarily on a disk file. After output completion, the entire output file is spilled contiguously to the output device. The printer and graphic plotter are incorporated in this subsystem. Also incorporated is one shared magnetic tape unit which allows the production of merged magnetic tapes for off-line communication with a larger computer center. A Spooled Input Subsystem had been determined to be non-essential for an Experiment Control Facility.

The <u>Spooled Datasink/Datasource Subsystem</u> manages on-line communication between a computer center and the Experiment Control Facility. This subsystem is specific to the protocol of the larger computer center and is therefore not appropriate to discuss here without describing the larger computer center.

The <u>File Subsystem</u> includes the File Management*** and the devices which contain on-line files.****

(a) In dynamically allocated files, space is allocated as needed by WRITE statements. This allocation algorithm is used with sequentially organized files.

(b) In preallocated files, the total space required is declared and allocated at the time of FILE CREATION. This file type is preferred for sequentially organized files with predictable sizes and for files with random access organizations.

(c) The object program library files include the libraries for overlays. This additional file type has been defined because of the difference in the access control flow between files containing programs and files containing data.

Access to the Central Shared Service Node

The Private Control Nodes access the Central Node via <u>Central Logical Channels</u> (CLC's). A group of channels is privately assigned to each connected Control Node. The Control Node may assign one dataset to a CLC. Thus the number of channels assigned to a given

***The File Management[13] as here defined manages the space allocation, file organization and retrieval of logical blocks. The File Management does not have knowledge of the <u>content</u> represented by bit patterns in the logical block; this would be the task of a Data Management.

****On-line files are those files having names which are interpretable and are recognizable by the Experiment Control Facility. Each on-line file thus has to be explicitly created. At this time at least a directory entry is generated. When no further use of an on-line file is necessary, the directory entry is explicitly destroyed.

Control Node determines the number of datasets to which the Control Node can connect concurrently.

A Control Node requiring a service initiates a request/response cycle, here called a Transaction. A transaction consists of three phases:

the REQUEST phase

the ACKNOWLEDGE phase

the DATA phase.

During the REQUEST phase, a 16-word logical block is transmitted from a Control Node to the Central Node. The content of the block includes the number of the CLC to be accessed as well as a complete description of the function requested from the dataset assigned to the CLC. Typical functions are:

CREATE/DESTROY file

RESERVE/RELEASE dataset

OPEN/CLOSE dataset

READ/WRITE logical block
to/from OPENed dataset

LOAD OVERLAY from Library.

The boundary protocols of these functions are standard within traditional input/output and file management systems and promise to be fairly stable.

At the ACKNOWLEDGE phase, the Central Node transmits the response back to the originating Control Node in a similar 16-word block. If the object of the request was a data transfer of some sort, then this transfer occurs at the DATA phase.

Malfunctions at the transaction level are reported during the ACKNOWLEDGE phase, while malfunctions at the hardware level are handled at the appropriate individual phase. Malfunctions occuring subsequent to some WRITE* transaction generate a message in a direction opposite to the Main Control Flow. These messages are considered "Unsolicited Responses" to a previously given transaction and are transmitted as transactions to the appropriate Control Node.

Hardware Characteristics of the Link

A hardware finger structure is used for the link between the Central Node and each Control Node. Each finger consists of a Central Node terminal, a Control Node terminal and an optically isolated multiwire cable. The sequence of control of each finger has the scope of one transaction. In order to allow for the transaction of "Unsolicited Responses" which are, as far as the hardware is concerned, initiated at the Central Node, the link control is symmetric and employs a master/slave organization.

*Some WRITE operations initialize a store-and-forward mechanism in the Central Node. Hence, they may generate malfunction messages which are out of time sequence with the originating program.

Flow of a Typical Transaction

Figure 7 shows a typical flow of a transaction using as an example a "WRITE LOGICAL BLOCK OF N WORDS TO A DATASET". Five interactions between the two nodes are necessary in a complete transaction. The Control Node obtains Master Status of the communication link and thus originates the transaction. It also assumes the responsibility for terminating the transaction with a release of the link. The Central Node cannot originate a transaction during the time the Control Node holds Master Status. Transactions between the other Control Nodes and the Central Node will not be disturbed.

I. Experiment Service Subsystem

The Experiment Service Subsystem is here defined to be a set of functional subsystems assigned between the Application Subsystem and the experiment electronics. All sorts of changes may happen in this area. Here, it is particularly difficult to find system designs which are incrementally expandable with prevailing technology.

Developments and partitioning in LSI technology have probably the most unsettling influence here.[14] Let me elaborate on the impact of the partitioning question on a simple historical example where the task is "MEDIUM SPEED (10^5 total counts/sec) SCALING FROM MANY SOURCES WITH ASSOCIATED ANALYSIS".

Figure 8 shows two implementations of the functional sequence SENSE, DERANDOMIZE, INCREMENT, STORE PREVIOUS COUNT, MULTIPLEX, ANALYZE. The two implementations correspond to designs which were practical in 1964[15,16] and in 1969.[17] In the 1964 design a pyramid hardware structure was used for interfacing between a computer on top of the pyramid and a single module level on the bottom of the pyramid. The functions DERANDOMIZE, INCREMENT, and STORE PREVIOUS COUNT were assigned to a module, which in turn was assigned to one detector. The function MULTIPLEX was scattered among controllers in the pyramid, computer hardware structures, and the computer software structures. The basic justification for this kind of distribution was that resources were expensive as compared to interconnections. Thus, resources should be shared wherever feasible at the expense of interconnections and explicit controllers. Around 1968 the appearance on the market of integrated, alterable diode matrices[17,18] made multiplexing from many sources a relatively inexpensive and fast function. This event in connection with the fact that core memory words were at this time already substantially more economical than individual scalers made the pyramid hardware structure obsolete for this task in all its structures, architectures, and subsystems as well as in a fair amount of its functional boundaries. The new design called for minimizing interconnections and explicit controllers at the expense of resources. Functional Subsystems were interconnected and distributed differently. MULTIPLEX moved to the hierarchy level closest to the detectors and DERANDOMIZE now became an explicit subsystem. INCREMENT followed on the next level and STORE PREVIOUS COUNT was accomplished in n memory locations within a dual port core memory. Software structures of the old design were eliminated. It would have been, obviously, a severe mistake at that time to proceed with refining the pyramid structure for that particular task.** Even to stick to

**There is ample evidence that in numerous experiment control applications, real-time applications and other computer applications conclusions pointing to similar trends have been reached.

the functional boundaries at that time would have been a serious mistake.

A projection to 1973 would show that:

(a) integrated diode matrices are today available in all sizes and shapes;

(b) the derandomizing buffer, at that time the most advanced and expensive development in the functional set, is today available as an integrated chip;[19]

(c) increment processors perform today, within the same global functional boundaries, more ambitious tasks such as matching events with windows, etc.;

(d) dual port memories are explicitly available even in small computers and implicitly standard as direct memory access channels.

The following points are noteworthy. Functional boundaries seem to stabilize. The distribution of functional subsystems among hardware and software subsystems and structures seems to be in a state of flux. The traditional concept of "interfacing electronics to a computer" is based on stable structures and thus adherence to this concept often cannot be justified later in terms of investment cost and depreciation cost.* A generally applicable modular approach with a reasonably useful life expectancy has to take functional boundaries and architectural interconnections into consideration.

Node-Memory Resource-Switch-Architecture

On the basis of the previous example, let me introduce in Fig. 9 a definition of the elements of an Experiment Service Subsystem and its architectural interconnections. It is called, for lack of another name, the NMS (node-memory resource-switch) definition or architecture of the Experiment Service Subsystem. The basic elements of the architecture are nodes, incomplete nodes, data switches, control switches, switched memory resources, and busses, where the busses have several levels of modular structure.

A node has four functionally defined ports: DATA OUT, DATA IN, GLOBAL CONTROL, LOCAL CONTROL. Functional ports may be missing or they may share the same hardware structure.

In the example of Fig. 9, individual lines from the detector electronics are connected to the DATA IN port of the Multiplexer Node. The multiplexed data are then routed via individual busses through the Derandomize Node, the Increment Node, and via a Data Switch to the Switched Memory Resource, which happens to be in this example a core memory. The data flow from the experiment to the Memory Resource is globally controlled by the higher level Control Node (the computer in the traditional description) using the Global Control port of the Multiplexer Node. It can be seen from the lack of control port connections, that the other two nodes contain looping processors.

*The time intervals over which these changes occur are highly environment specific.

The Display Node is assumed to contain a processor which formats data from the memory resource for the purpose of point plotting on a X-Y-Z scope. As shown, the display may be controlled globally from the Control Node or the control lines may be switched to a manual display function board. In addition, an intermixed control is possible.** Busses 1 and 2 are on lower levels of modular structure. Busses 3 and 4 are multidirectional and thus obey a complete master/slave control protocol.

Incomplete Nodes

As pointed out previously, the most fragile structures within an Experiment Service Subsystem seem to be those connected with software. Many of these structures will be partially or totally eliminated by rapid technological advances. In order to satisfy the incremental expandability requirement it is necessary to maintain at all levels of implementation functional boundaries even if present conditions call for complex scattering between several software structures and several hardware structures. If these boundaries are maintained, a good probability exists that such functions can be implemented later in different partitionings by elimination or depopulation of existing structures and not by modification (modifications create new interfaces). Such a strategy keeps the structures technologically useful as long as the structures which are depopulated but unremoveable do not represent a major part of the existing system.

The hardware interfaces of the scattered functional subsystems are referred to as Incomplete Nodes. The software subsystems operate in the Control Node in conjunction with the Application Subsystem.

Hardware Bus Structure

As the basic hardware, the UNIBUS[20,21] has been selected. It satisfies the requirements of a modern hardware structure:

(a) minimum restrictions for the node hardware structure and mechanical measurements;

(b) uncommitted data and control flow direction within a bus (master/slave control organization);

(c) uncommitted interconnection architecture for the connection of multiple busses to nodes;

(d) provision for separation between data and control flow;

(e) request/response relationship of the control and data flow at all levels.

**For instance, the following functions could be implemented. Move several parameters from the control node to the display node. Select the parameters presently to be used via a local function board.

The disadvantage, of course, is that this bus hardware is a proprietary product of one company. However, of the available modern bus structures, the UNIBUS seems to enjoy a widespread acceptance at this time. Thus, a reasonable probability exists of finding a large selection of compatible hardware on the market.

J. Remarks on the Implementation

Computer Hardware

One PDP11/40 [20] each will be assigned to the Central Shared Service Node and to the eleven presently planned Private Control Nodes. Ten of these Control Nodes will be connected to individual experiments. The remaining nodes will be used for program preparation, since it may be necessary to do program preparation in parallel with an experiment run. The same processor for all nodes was chosen for reasons of interchangeability in the case of maintenance problems and for the simplicity of initial programming. The concept, however, does not require the same processor at all control nodes.

Appendix 3 lists the presently planned hardware configuration.

Software

It is necessary in this project to concentrate more on the structure of the software and the boundaries of the subsystems than on the features of the various functions. It has been taken into consideration that existing software boundaries are often a curious mixture between conceptual intentions and violations resulting from realities. A study of the structures reveals that generally:

- Boundaries between application programs (software processors) and their run-time systems seem to be reasonably clean;
- boundaries between the run-time system and the first level of the operating system seem to be vague;
- boundaries within the operating system are often well-defined on the conceptual level. On the implementation level, these definitions are often severely violated.

Some important points for the partitioning task are as follows:

- Boundaries between company supplied software processors and the first level of the operating system are kept intact and, if necessary, simulated. We hope to be able to use future versions of company supplied processors. A continuous interface effort is expected here which has an impact on the operating cost of the facility.
- The operating system is partitioned according to the functional assignments outlined in the presentation. Functionally recognizable portions are kept; missing parts are coded. Unnecessary parts are removed.
- The Central Shared Service Node contains, expressed in traditional terms, only system software. The resource sharing operating system software is expected to stabilize. Additions to the subsystems are expected in the future; modifications are unlikely. In order to accommodate this philosophy, the software for the Central Node is designed essentially from scratch.
- Software for the Experiment Service Subsystem is initially complex but is expected to decrease in complexity and volume in the future by depopulation as a result of scheduled enhancement of the appropriate experiment or laboratory setup with state-of-the-art technology. Thus, a serious attempt has been made to find a structure which will allow depopulation without change.

Schedule

We expect to have the system in operation by the end of 1974.

K. Acknowledgements

The author wishes to thank V. Radeka for very helpful guidance and advice. Extensive discussions about the concepts with S. Rankowitz and F. Stubblefield are gratefully acknowledged.

Appendix A

Remarks on Design Parameters:

1. Parameters with primary impact on investment cost

Economically achievable response time

There is a direct relationship between response time requirements and investment cost in resource sharing systems. The longer the response time requirements, the more efficiently available resources can be managed. Elaboration on this parameter is given elsewhere.[12]

Transferability of functional subsystems

The degree to which functional subsystems are applicable in and transferable to other installations in the experiment control field, to related fields and to unrelated fields in the general purpose computer area is expressed with this parameter.

Relative hardware investment cost

The relative distribution of hardware resources within a typical experiment control system is considered here. In the determination of the parameter properties, consideration has been given to the fact that electromechanical resources such as peripherals and file devices are likely to reduce substantially less in cost than purely electronic resources such as processors and memory. The cost of the latter resources is determined by the economy achieved in large-scale integration and other electronic fields. For instance, the cost of a core memory bit has decreased approximately one order of magnitude in five years.

2. Parameters with primary impact on operating cost

Mean time between changes

Changes have a considerable impact on the operating cost of a subsystem. This is particularly true for functional subsystems which include:

(a) complex system software;

(b) inflexible hardware interconnections.

The operating cost is mainly generated by:

1. the hardware development cost generated by the change;
2. the software development cost generated by the change;
3. the obsolete hardware and software due to the change;
4. the introduction of unsettling new operating procedures and new design errors. This point has, as a matter of experience, a high influence if software changes are necessary.

Damage caused by unisolated malfunctions

This aspect of system reliability has often been overlooked. For instance, a parity error in primary memory within the Experiment Service Subsystem is a relatively minor malfunction but it can cause substantial damage if reinitialization of an "operating system" is necessary and this operation results in the loss of temporary files. The latter serious damage results because the consequences of the malfunction were not restricted to the functional subsystem where the malfunction occurred. It is interesting to note that this kind of behavior of a computer system is often taken for granted or at most meets minor opposition. Elaboration on the definition of malfunction isolation is given elsewhere.[10]

3. Parameters with primary impact on depreciation

Technical obsolescence rate of functional subsystems

A functional subsystem is considered obsolete if the cost/performance ratio of the function is at a level such that it would no longer make sense to construct such a design. The obsolescence of functional subsystem boundaries is a different matter which relates to obsolescence of the architecture of the entire Experiment Control System.

Appendix B

Presently Planned Hardware
Configuration of Reactor Experiment Control Facility

1. Central Shared Service Node consists of:

Common Equipment

PDP11/40 processor

48K core memory

Keyboard/printer terminal

CRT terminal

File Subsystem

0.5 million words fixed head disc,
8 msec average access, 4 μsec/word transfer

20 million words disc pack unit
30 msec average seek time, 7.5 μsec/word transfer

Reserved Device Subsystem

Two 9-track magnetic tape units

Card reader

Paper tape reader/punch

DEC tape

Spooled Output Device Subsystem

7-track magnetic tape unit

Electrostatic line printer

Electrostatic plotter

2. Eleven Control Nodes each consisting of:

PDP11/40 processor

16K core memory

Keyboard/printer terminal

Paper tape reader (for bootstrap and "minimum stand-alone" operations).

3. Communication Link Between Controlled Nodes and Central Nodes:

Concept, development and implementation done at BNL. The maximum data rate at each finger of the link is ~ 100K words (16 bits/word).

References

1. Session "Data Processing in Nuclear Electronics" in Proc. of Int. Symp. on Nucl. Electr., Paris, November 1963.

2. Proc. of Conf. on Automatic Acquisition and Reduction of Nuclear Data, Karlsruhe, July 1964.

3. Proc. of Conf. on Comp. in Exp. Physics, Skytop, March 1969.

4. L. J. Lidofsky, Contrast and Similarities in Systems; Proc. on Future Trends in Nuclear Instrumentation, Ispra, May 1969.

5. D. G. Dimmler. Panel discussion on Proc. of Conf. on Comp. in Exp. Physics, p. 178, Skytop, March 1969.

6. P. R. Beaucage, M. A. Kelley, D. Ophir, S. Rankowitz, R. J. Spinrad and R. V. Norton. Multiple Spectrometer Control by Computer, Nucl. Inst. and Methods 40(1966) 26-44.

7. L. Stanchi. Report on Panel Discussion; Proc. on Future Trends in Nuclear Instrumentation, Ispra, May 1969.

8. H. Chestnut. System Engineering Tools, p. 12, J. Wiley & Sons, New York 1966.

9. J. M. Salzer. Evolutionary Design of Complex Systems. In D. P. Eckman Systems: Research and Design, J. Wiley & Sons, New York 1961.

10. D. G. Dimmler. Architecture of a Maintenance Subsystem in a Multi-User Computer Installation. Proc. of Workshop on Fault Detection and Diagnosis in Digital Circuits and Systems, Lehigh University, Allentown, December 1970.

11. TTL Databook, Texas Instruments, 71241-23-CHI, p. 398, 1973.

12. D. G. Dimmler. Study for the Expansion of the On-line Data Processing Facility. BNL 50180 (T-536), BNL external report, June 1968.

13. A. J. Collmeyer. File Organization Techniques. Computer Group News, Vol. 3, No. 2, March 1970.

14. Fairchild, OPTIMOS-MOS Technology, p. 9, Fairchild Semiconductor, Mountain View, California, September 1972.

15. G. Krüger and D. G. Dimmler. A Multiple Input Data Acquisition System Using a Small On-line Computer. Proc. of Conf. on Automatic Acquisition and Reduction of Nucl. Data, Karlsruhe, July 1964.

16. D. G. Dimmler and G. Krüger. A Display System With an On-line Computer, Proc. of Conf. on Automatic Acquisition and Reduction of Nucl. Data, Karlsruhe, July 1964.

17. D. G. Dimmler. A Computerized Data Acquisition System for High Event Rates from Many Sources. Proc. Nuclear Electronics Symposium, Ispra, May 1969.

18. Radiation, Inc., Microelectronics Division, Melbourne, Florida.

19. Signetics, Digital Circuits, 1972, p. 7-135.

20. PDP11/40 Processor Handbook, Digital Equipment Corporation, Maynard 1972.

21. DIGITAL LOGIC HANDBOOK, Digital Equipment Corporation, Maynard, 1973.

<u>Figure Captions</u>

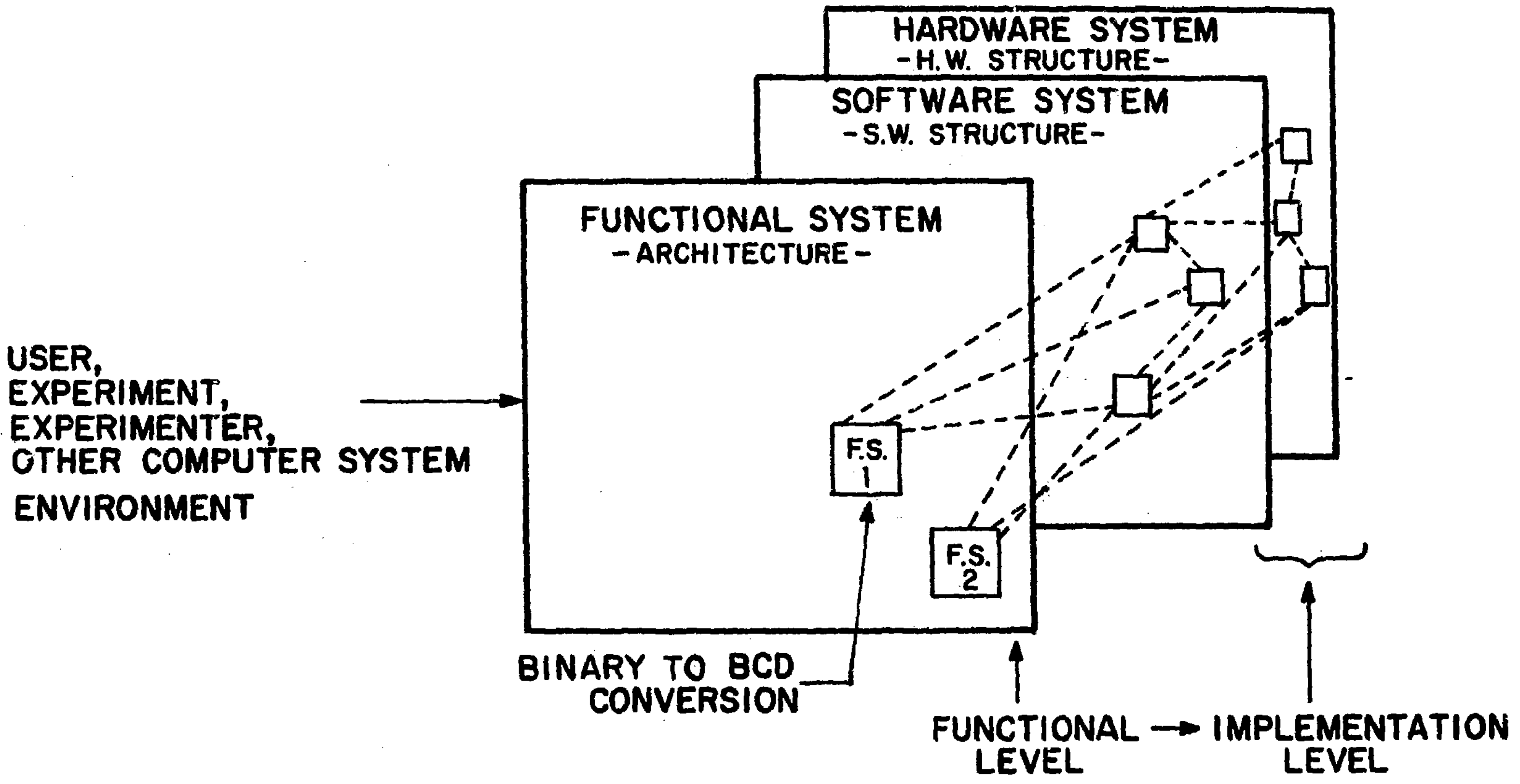

HARDWARE SYSTEM
-H.W. STRUCTURE-
SOFTWARE SYSTEM
-S.W. STRUCTURE-
FUNCTIONAL SYSTEM
-ARCHITECTURE-
USER,
EXPERIMENT,
EXPERIMENTER,
OTHER COMPUTER SYSTEM
ENVIRONMENT
F.S. 1
F.S. 2
BINARY TO BCD CONVERSION
FUNCTIONAL LEVEL
IMPLEMENTATION LEVEL

FIGURE 1

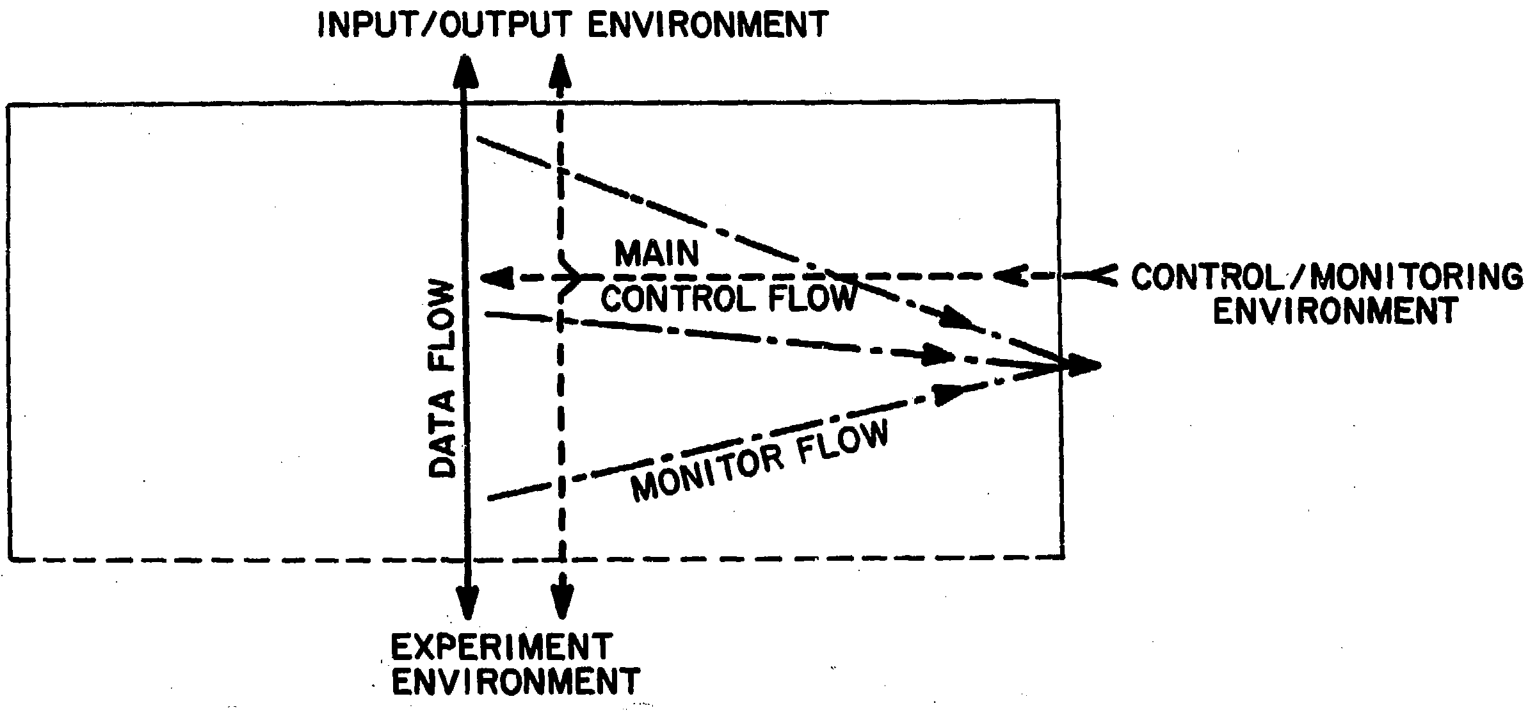

INPUT/OUTPUT ENVIRONMENT
DATA FLOW
MAIN CONTROL FLOW
MONITOR FLOW
CONTROL/MONITORING ENVIRONMENT
EXPERIMENT ENVIRONMENT

FIGURE 2

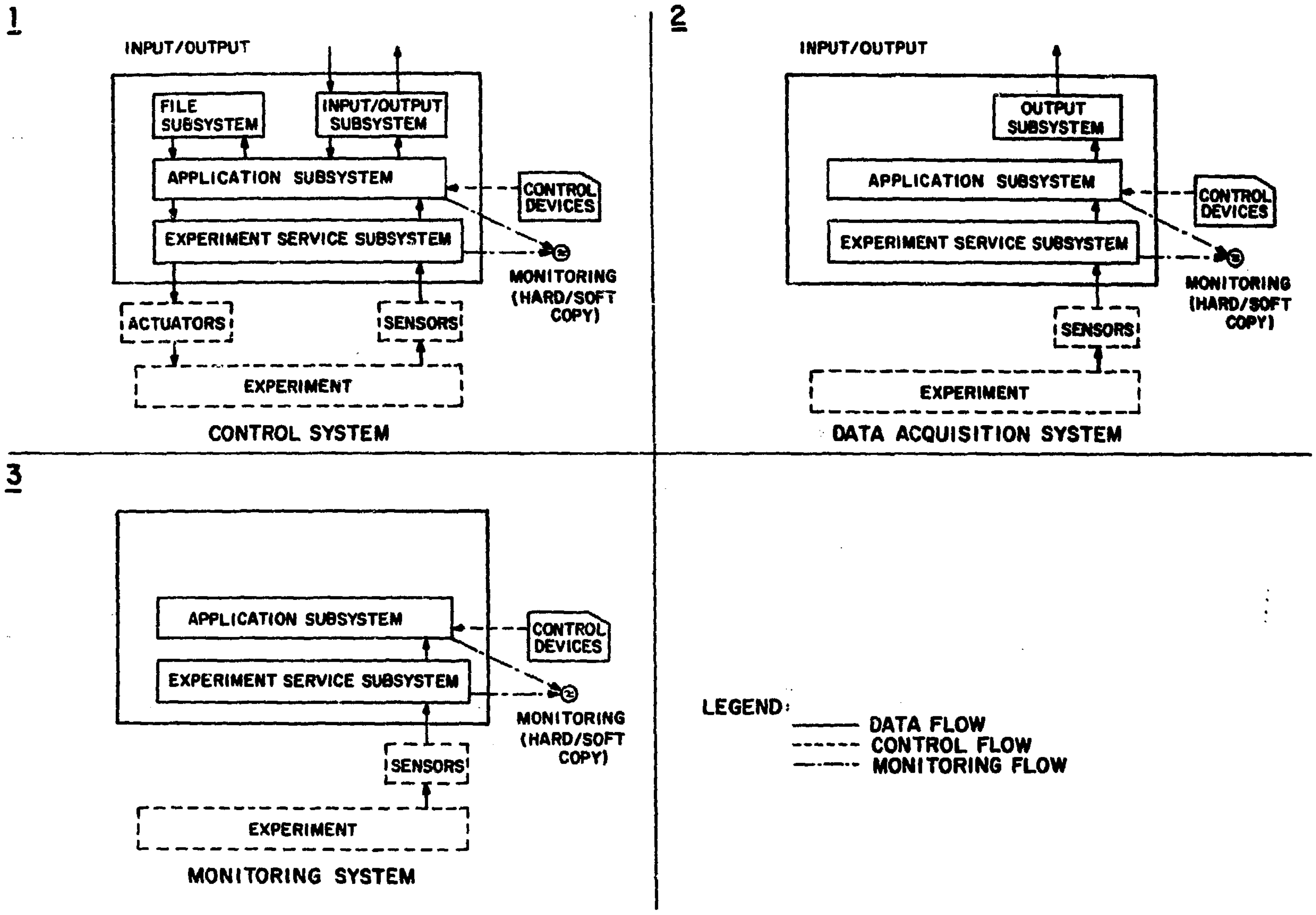

1
INPUT/OUTPUT
FILE SUBSYSTEM
INPUT/OUTPUT SUBSYSTEM
APPLICATION SUBSYSTEM
CONTROL DEVICES
EXPERIMENT SERVICE SUBSYSTEM
MONITORING (HARD/SOFT COPY)
ACTUATORS
SENSORS
EXPERIMENT
CONTROL SYSTEM
2
INPUT/OUTPUT
OUTPUT SUBSYSTEM
APPLICATION SUBSYSTEM
CONTROL DEVICES
EXPERIMENT SERVICE SUBSYSTEM
MONITORING (HARD/SOFT COPY)
SENSORS
EXPERIMENT
DATA ACQUISITION SYSTEM
3
APPLICATION SUBSYSTEM
CONTROL DEVICES
EXPERIMENT SERVICE SUBSYSTEM
MONITORING (HARD/SOFT COPY)
SENSORS
EXPERIMENT
MONITORING SYSTEM
LEGEND:
DATA FLOW
CONTROL FLOW
MONITORING FLOW

FIGURE 3

Design parameter properties / Subsystems	INVESTMENT COST			OPERATING COST		DEPRECIATION COST	REASON FOR SHARING OF SUBSYSTEM BETWEEN SEVERAL EXPERIMENTS
	Economically achievable response time	Transferability of <u>functional</u> subsystem	<u>Relative</u> Hardware Investment Cost	Mean Time Between Changes MTBC	Damage caused by unisolated malfunction	Technical obsolescence rate of functional subsystem (<u>not</u> functional boundaries)	
INPUT/OUTPUT SUBSYSTEM	> 10^1 sec	Generally applicable to fields unrelated to experiment control	High, particularly for I/O devices and file devices	Long	High (damage to file unrelated to malfunction)	Low > 5 years	1. Economic 1.1 High investment in I/O devices 1.2 Low/medium usage of devices
FILE SUBSYSTEM	~ 5 x 10^{-2} to 10^{-1} sec				Very high (damage to reference file unrelated to malfunction)	Very low > 5 years	1. Same as above 2. Sharing of reference files and libraries (data sharing)
APPLICATION SUBSYSTEM (Excluding Application Program)	~ 10^{-3} sec		Medium	Medium	Medium	Medium ~ 2 to 3 years	Economy of processors and primary memory
EXPERIMENT SERVICE SUBSYSTEM	~ 10^{-5} sec	Global subsystem specific, hopefully more elementary subsystems transferable to related fields	Low	Short	Low	High ~ 1 year	Economy of computer (primary memory + processor hardware + system software) technology as compared to other technologies (LSI digital, analog).
EXPERIMENT ELECTRONICS	~ 10^{-9} sec					High ≤ 1 year	

FIGURE 4

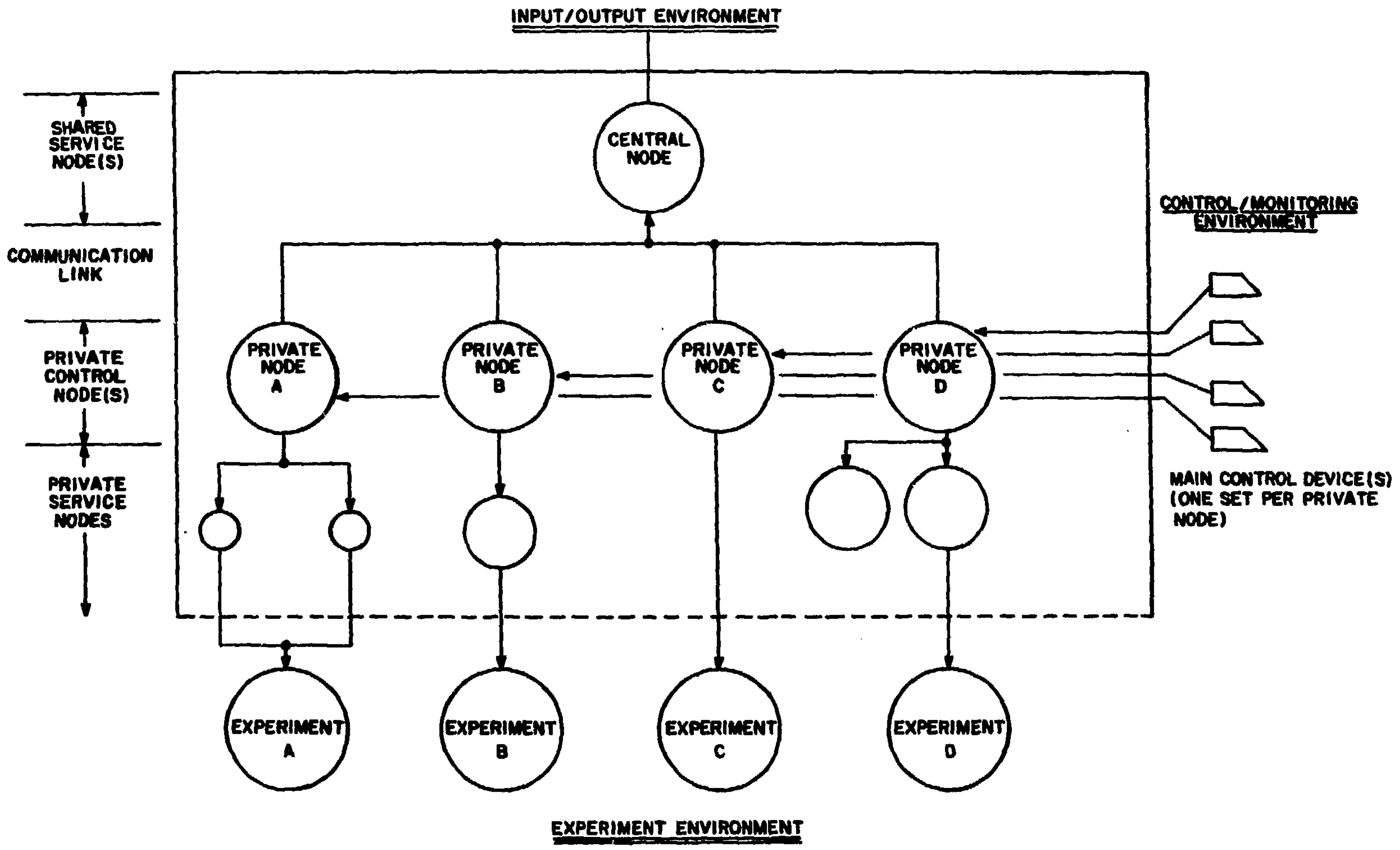
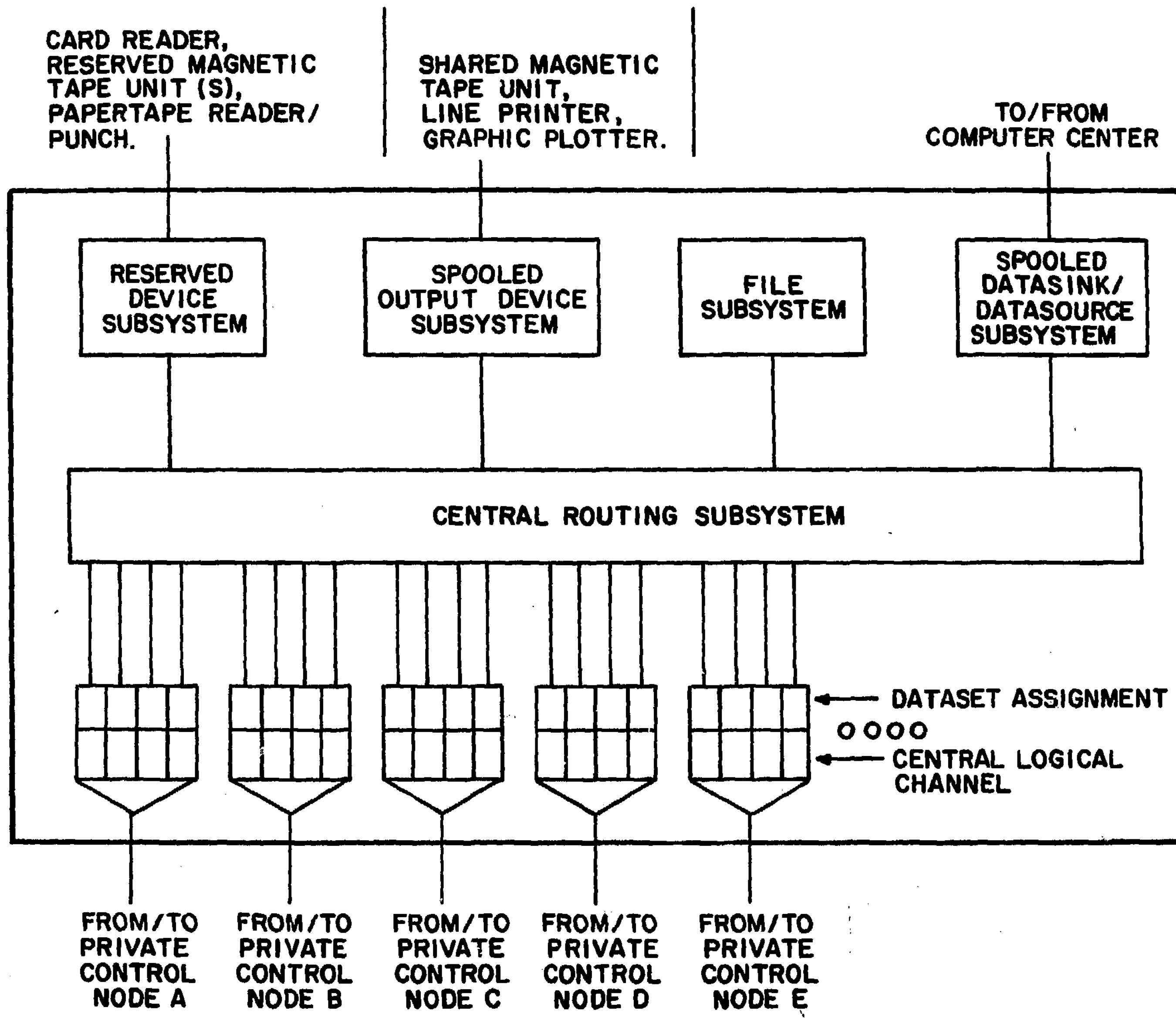

INPUT/OUTPUT ENVIRONMENT
SHARED SERVICE NODE(S)
COMMUNICATION LINK
PRIVATE CONTROL NODE(S)
PRIVATE SERVICE NODES
CENTRAL NODE
CONTROL/MONITORING ENVIRONMENT
PRIVATE NODE A
PRIVATE NODE B
PRIVATE NODE C
PRIVATE NODE D
MAIN CONTROL DEVICE(S) (ONE SET PER PRIVATE NODE)
EXPERIMENT A
EXPERIMENT B
EXPERIMENT C
EXPERIMENT D
EXPERIMENT ENVIRONMENT

FIGURE 5



CARD READER, RESERVED MAGNETIC TAPE UNIT (S), PAPERTAPE READER/ PUNCH.
SHARED MAGNETIC TAPE UNIT, LINE PRINTER, GRAPHIC PLOTTER.
TO/FROM COMPUTER CENTER
RESERVED DEVICE SUBSYSTEM
SPOOLED OUTPUT DEVICE SUBSYSTEM
FILE SUBSYSTEM
SPOOLED DATASINK/ DATASOURCE SUBSYSTEM
CENTRAL ROUTING SUBSYSTEM
DATASET ASSIGNMENT
O O O O
CENTRAL LOGICAL CHANNEL
FROM/TO PRIVATE CONTROL NODE A
FROM/TO PRIVATE CONTROL NODE B
FROM/TO PRIVATE CONTROL NODE C
FROM/TO PRIVATE CONTROL NODE D
FROM/TO PRIVATE CONTROL NODE E

FIGURE 6

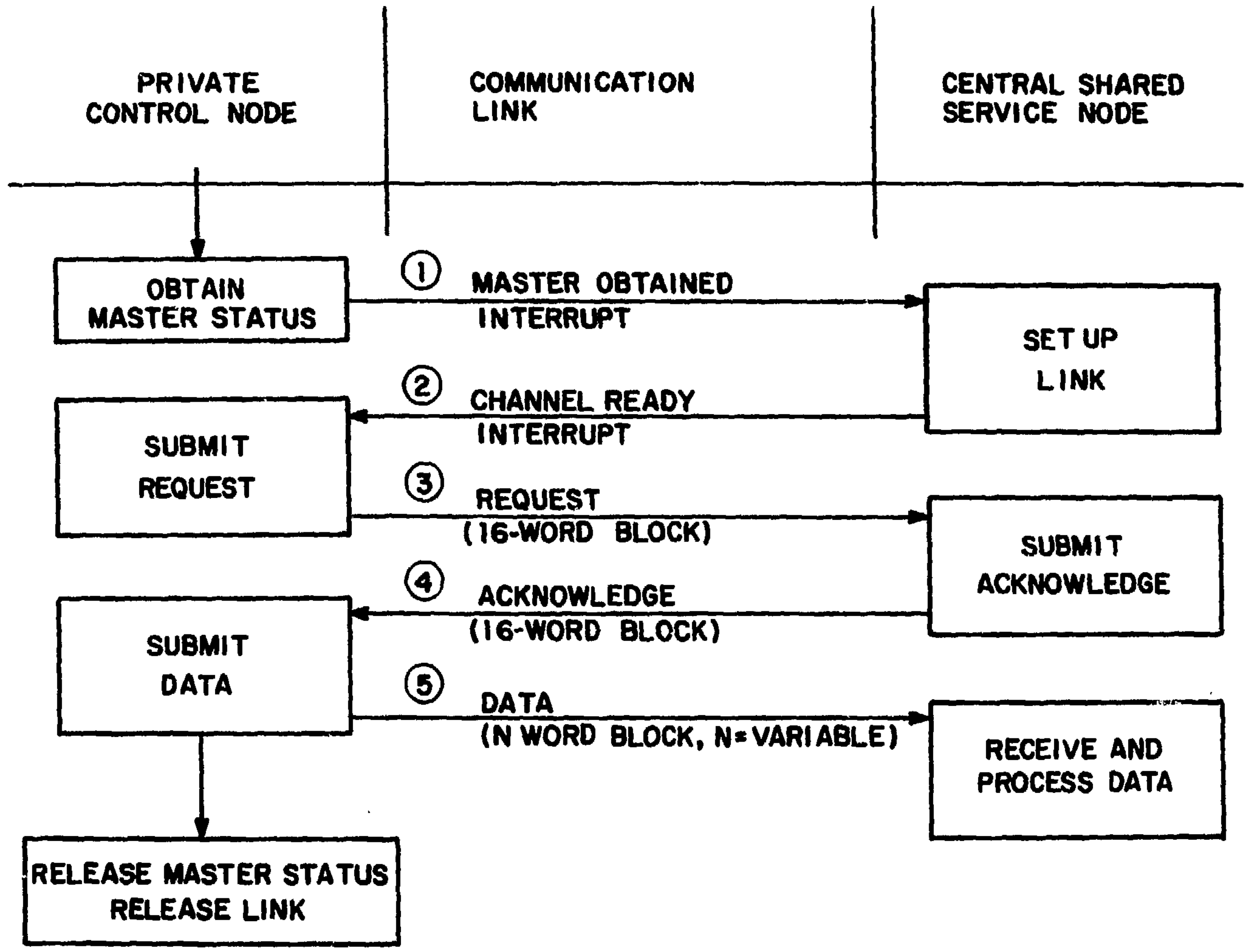
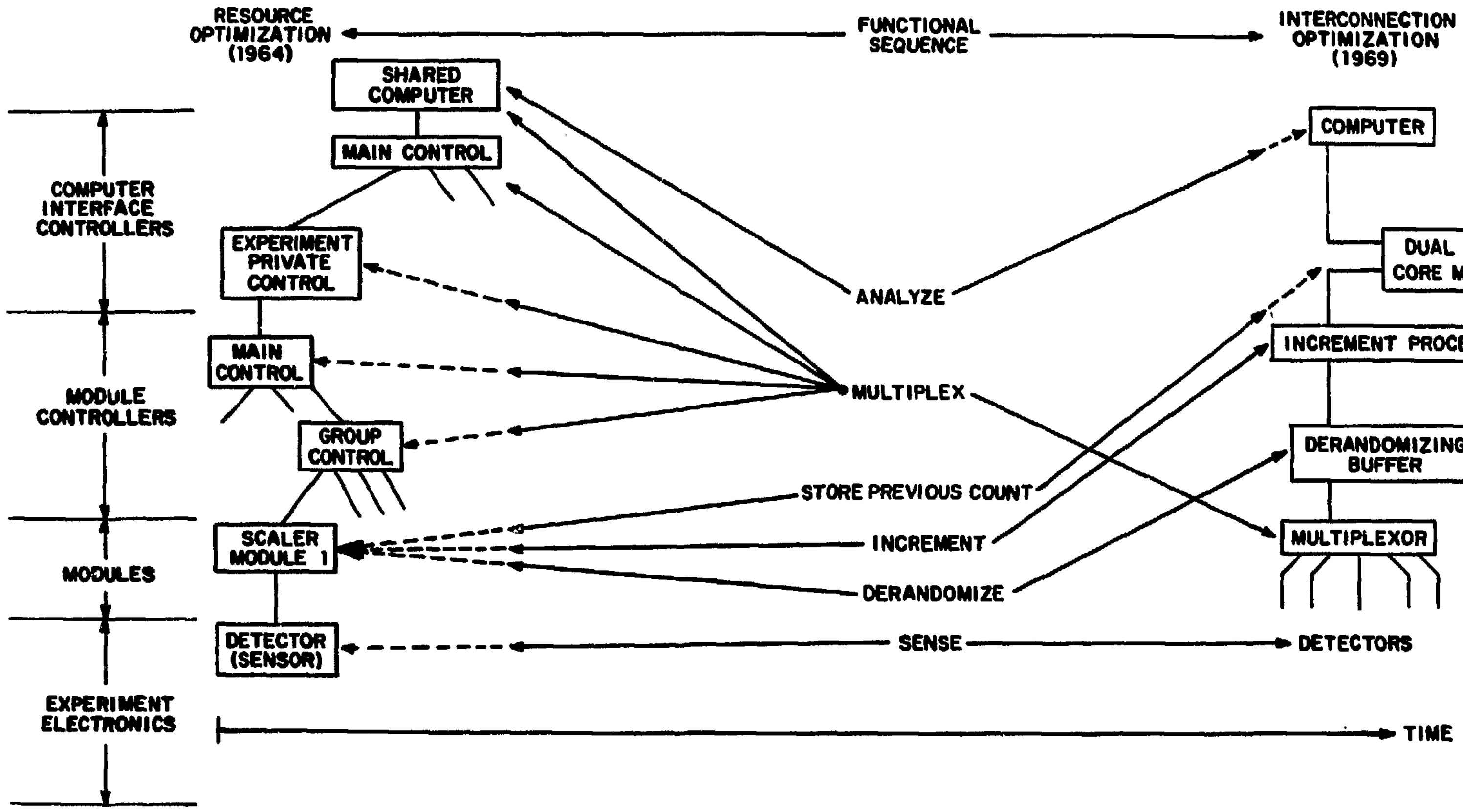

PRIVATE CONTROL NODE
COMMUNICATION LINK
CENTRAL SHARED SERVICE NODE
OBTAIN MASTER STATUS
① MASTER OBTAINED INTERRUPT
SET UP LINK
② CHANNEL READY INTERRUPT
SUBMIT REQUEST
③ REQUEST (16-WORD BLOCK)
SUBMIT ACKNOWLEDGE
④ ACKNOWLEDGE (16-WORD BLOCK)
SUBMIT DATA
⑤ DATA (N WORD BLOCK, N=VARIABLE)
RECEIVE AND PROCESS DATA
RELEASE MASTER STATUS RELEASE LINK

FIGURE 7



RESOURCE OPTIMIZATION (1964)
FUNCTIONAL SEQUENCE
INTERCONNECTION OPTIMIZATION (1969)
COMPUTER INTERFACE CONTROLLERS
MODULE CONTROLLERS
MODULES
EXPERIMENT ELECTRONICS
SHARED COMPUTER
MAIN CONTROL
EXPERIMENT PRIVATE CONTROL
MAIN CONTROL
GROUP CONTROL
SCALER MODULE 1
DETECTOR (SENSOR)
ANALYZE
MULTIPLEX
STORE PREVIOUS COUNT
INCREMENT
DERANDOMIZE
SENSE
COMPUTER
DUAL PORT CORE MEMORY
INCREMENT PROCESSOR
DERANDOMIZING BUFFER
MULTIPLEXOR
DETECTORS
TIME

FIGURE 8

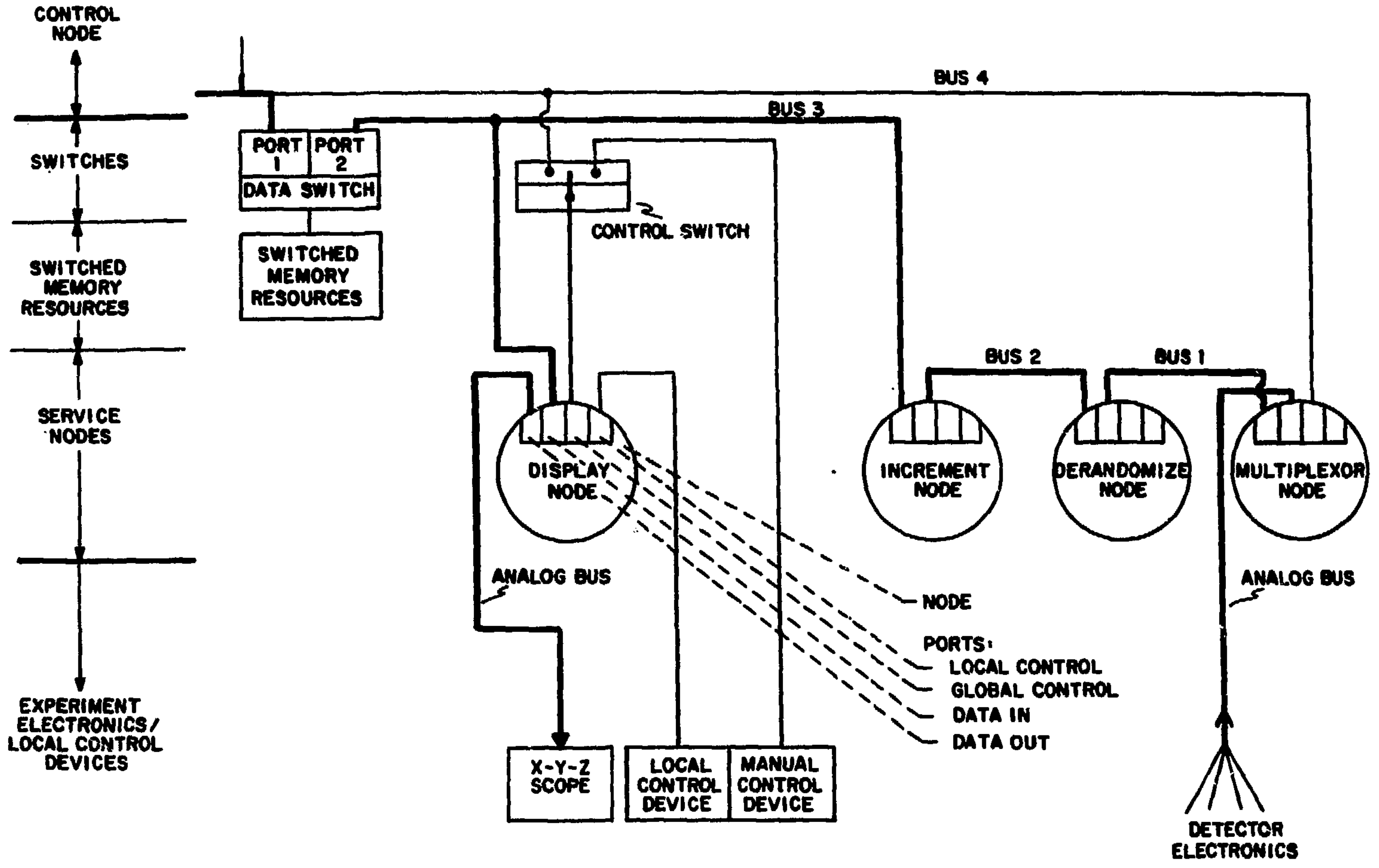

CONTROL NODE
SWITCHES
SWITCHED MEMORY RESOURCES
SERVICE NODES
EXPERIMENT ELECTRONICS/ LOCAL CONTROL DEVICES
BUS 4
BUS 3
PORT 1
PORT 2
DATA SWITCH
SWITCHED MEMORY RESOURCES
CONTROL SWITCH
BUS 2
BUS 1
DISPLAY NODE
INCREMENT NODE
DERANDOMIZE NODE
MULTIPLEXOR NODE
ANALOG BUS
ANALOG BUS
NODE
PORTS:
LOCAL CONTROL
GLOBAL CONTROL
DATA IN
DATA OUT
X-Y-Z SCOPE
LOCAL CONTROL DEVICE
MANUAL CONTROL DEVICE
DETECTOR ELECTRONICS

FIGURE 9